Comparison and Development of Resource Efficient Machine Learning Models

Abhishek Kumar 18111002
Darshak Chhatbar 18111012
Deepak Yadav 18111014
Deepak Yadav 18111015
Nitin Vivek Bharti 18111048
Rahul Kumar Singh 18111052

Abstract

This project analyses some of the recent advances in machine learning field for resource constraint devices and proposes a novel convolutional approach to Bonsai Tree model. These models try to minimize the resource requirements like RAM and storage without hurting the accuracy much. Experimental results on multiple benchmark datasets show that our modification to bonsai tree further reduces the resource requirement than normal bonsai tree, exploits the local structure of the data and increases the explainability of convolutional networks.

1 Introduction

Objective

To analyse some of the recent advances in machine learning for resource scarce devices and to develop such machine learning models.

Motivation on Resource constraint devices and IoT

Devices like Arduino board, ATmega328 etc. are essential in IoT infrastructures like health care, smart grids, wearables, energy management etc. these devices transfer data to cloud to extract some information and are dependent on them.

These devices need models that can run without depending on cloud computing since cloud connectivity is not present everywhere, in a network data security can be compromised, it takes time to compute and transfer data which might not be good for real time analysis. So we need models that can run locally and do not require large resources and do not hurt the accuracy of the task much.

2 Literature Survey

There have been many advances in this field like reducing the prediction cost of KNN with prototype based methods like ProtoNN [1] and BNC(Binary Neighbour Compression) [2] where you learn prototypes to reduce model size, SNC(Stochastic Neighbour compression) [3] where you learn very small synthetic dataset to perform KNN, Tree based methods like bonsai tree where under constraints model learns non-linear decision rules at each node [4], pruning the random forests based on resource constraints [5], model compression [6] , parameter pruning and sharing for CNNs and Dense Networks [7] by reducing redundant parameters that are not sensitive to task, Low rank factorization of Neural nets [8] using matrix or tensor decomposition to estimate importance of parameters, Compact convolutional filters [9] by designing special convolution filters to save parameters. We have analyzed mainly three models ProtoNN, BNC and Bonsai Tree.

2.1 ProtoNN

ProtoNN[1] is kNN based model that uses compressed model and prototypes for prediction. Prototypes are learned from the the data along with the estimation of projection matrix jointly, due to which it avoids pruning after the model is learnt to fit the model in desired memory. Prototypes are points that represent the entire data. The projection matrix is sparse matrix estimated by performing SGD and iterative hard-thresholding. Since number of prototypes are far less than number of inputs and number of features are less the model is comparatively small. ProtoNN gives nearly the same accuracy as most popular models that take a huge amount of RAM with very small amount of memory used , which makes it fit for our use in resource constrained IoT devices.
ProtoNN tries to optimize the following loss function:

$$\mathcal{L}_i(Z,B,W) = \mathcal{L}(y_i, \sum_{j=1}^{m} z_j K_\gamma(b_j, Wx_i)) \tag{1}$$

This is for each data point i. B is prototypes and Z is its corresponding score vector. W is low-dimensional projection matrix. K_γ is RBF similarity kernel function used in the paper, any other kernel function can be used as well. The optimizing problem they obtained is non-convex but alternating optimization works in this case. Each of the parameters (B, Z, W) are learnt alternately using the algorithm provided with sparsity constraints.

2.2 BNC

Another simple model which is similar to the one described above is Binary Neighbor Compression (BNC). Here a KMeans clustering is performed on each class and number of clusters from each class are given by $k_y = \beta N_y$ where β ϵ (0,1) and N_y is the number of points that belong to class y. In this way we create a matrix of prototypes **C** of size m X d, where m is the total number of prototypes, and d is the dimension of the data.
Then we initialize a matrix W of size d X r randomly, where r is the new dimension of data. Using this we convert the Prototypes to lower dimension representation and also binary form as: $B = sign(CW)$. Hence B will be of the size m X r. After this we learn B and W alternately, using the loss function below which is similar to multi-class hinge loss:

$$\min_{W,B} \frac{1}{N} \sum_{i=1}^{N} [\alpha - \max_{j:z_j=y_i} (\tanh(\gamma W^T x_i)^T b_j) + \max_{j:z_j \neq y_i} (\tanh(\gamma W^T x_i)^T b_j)]_+ + \lambda \sum_{k \epsilon [r]} (||w_k||^2 - 1)^2$$

where tanh is used instead of sign function, as sign is not differentiable. So as $\gamma \rightarrow \infty, \tanh(\gamma W^T x_i) \rightarrow sign(W^T x_i)$.Here $[x]_+ = \max\{0, x\}$ and α is a hyper-parameter. Also a regularization is applied on W.
Prediction is made by computing the similarity of projected test point with all the prototypes, and the label of the prototype which has highest similarity is assigned to the test point.

2.3 Bonsai Tree

Unlike normal trees which learn axis aligned decision rules bonsai learns a non linear decision rule at every node. It first projects the data into low dimensional space(can be done in streamlined fashion), then projected features are traversed through the tree with each node scoring the output in their own way and sum of all scores is used as net score.

Scoring Function

Bonsai learns a single, shallow sparse tree whose predictions for a point x is given by :

$$y(x) = \sum_k I_k(x) W_k^T Zx \odot \tanh(\sigma V_k^T Zx)$$

where $\odot$ denotes the element wise Hadamard product, σ is a hyper-parameter, Z is a sparse projection matrix and $I_k(x)$ is an indicator function taking the value 1 if node k lies along the path traversed by x and 0 otherwise and W_k and V_k are sparse scoring vectors learnt at node k.

Branching Function

Bonsai tree computes I_k by learning a sparse vector θ_k at each internal node such that the sign of $\theta_k^T Zx$ determines whether data object x should be branched to the left or right child. Optimizing the I_k is hard problem so it is relaxed as follows:

$$I_{k>1} = 0.5 * I_j(x)(1 + (-1)^{k-2j} \tanh(\sigma_I \theta_j^T Zx))$$

where, j^{th} node is parent of k^{th} node.

Optimization Problem

The optimization problem can be formulated with any empirical loss function (e.g categorical cross entropy loss), as follows :

$$\min_{\Theta}\{\mathcal{L}(y, x, \Theta) + \frac{\lambda_\theta}{2}Tr(\theta^T\theta) + \frac{\lambda_W}{2}Tr(W^TW) + \frac{\lambda_V}{2}Tr(V^TV) + \frac{\lambda_Z}{2}Tr(ZZ^T)\}$$

All the parameters are simultaneously optimized in alternating fashion. This model now can be trained using gradient descent approach, newton method etc. the original implementation used gradient descent with IHT(iterative hard thresholding) constraints over the parameters.

3 ConvBonsai : Proposed Convolutional Bonsai Model

We introduce a new modification to the bonsai tree model by replacing the projection matrix Z with a Convolution Tree like model. Here each internal node is a convolutional layer with small number of filters (4-5). There's no scoring on output of convolutional layers but branching is done. At every leaf of convolution tree we have a small bonsai tree model that classifies the object instance.

Scoring function

Now scoring function tree gets modified as follows :

$$y(x) = \sum_m I_m(\sum_k I_{km}(x)W_{km}^T C_m(x) \odot \tanh(\sigma_m V_{km}^T C_m(x))$$

where outer sum is only over leaf nodes of convolution tree and I_m is the indicator function that object is traversing through that leaf node, $C_m(X)$ is the output after convolution at leaf node m, and $I_{km}(x)$ is an indicator function taking the value 1 if node k lies along the path traversed by x and 0 otherwise and W_{km} and V_{km} are sparse scoring vectors learnt at node k for bonsai tree located at leaf node m of convolution tree.

Branching function convolution tree

Similar to bonsai tree branching is done with help of a learnable parameter θ in following relaxed manner:

$$I_{k>1} = 0.5 * I_j(x)(1 + (-1)^{k-2j} \tanh(\sigma_I \theta_j^T C_j(x)))$$

where, node j is parent of node k and $C_j(x)$ is convolution output of k's parent node (i.e we compute child layer output in following manner):

$$C_k(x)_{ij}^r = g((W_r * C_j(x))_{ij} + b_r)$$

where, W_r, b_r are convolution filter weights and biases for filter r, g is activation function
Branching in bonsai tree at leaf remains as it is for transformed input.

Optimization Problem

The optimization problem can be formulated in similar way as:

$$\min_{\Theta}\{\mathcal{L}(y,x,\Theta)+\frac{\lambda_\theta}{2}Tr(\theta^T\theta)+\frac{\lambda_W}{2}Tr(W^TW)+\frac{\lambda_V}{2}Tr(V^TV)+\sum_c\frac{\lambda_c}{2}Tr(W_cW_c^T)\}$$

where, W_c are filter weights of each convolution filter and now, This model was trained using gradient descent approach with IHT(iterative hard thresholding) constraints over the parameters.

This has several advantages over normal bonsai tree such as:

- It exploits the local structure in data.
- It further reduces the number of parameters required.
- Convolutional outputs are more explainable here than conventional CNNs as observed in results, since the classification task gets split into sub-classification task with similar classes in a set.
- Compresses a CNN like model for low resource devices.
- Sequential Learning is possible. i.e(If we train again on a new class it can retain the properties learned for old classes) since the training is done only for sub part of tree all other gradients becomes zero.

Pooling layers can also be introduced to further decrease number of parameters. While training σ_m is kept small and eventually increased to avoid bad local minima(skewed sparse vectors for parameters) of optimization problem.

4 Tools and Software Used

Automatic differentiation tools were required to train the models so, all the models were modeled in **python** with **Tensorflow framework** and optimizer used were **Gradient Descent optimization** and **Adam optimization**. Some standard models like logistic regression, KNN, RFC were also analyzed in terms of memory requirement with help of **sci-kit learn** library in python. Code of ConvBonsai adn other models that we implemented can be found in references [13], [14].

5 Experimental Results

Datasets : Experiments were done on publicly available standard datasets for binary and multiclass classification that includes, Cifar-10, Mnist-10, USPS-1, notMnist, Iris, Madelon. some stats about these datasets:

Dataset	#Train	#Test	#Features	#Classes
Mnist-10	60,000	10,000	784	10
USPS-1	7291	2007	256	10
Cifar-10	50,000	10,000	3072	10
notMnist	75,000	25,000	784	10
Madelon	2600	1800	500	2
Iris	125	25	4	3

Table 1: Stats of datasets used.

Hyperparameter Tuning : We splitted the training data into 80-20 percent for training and validating based on class labels(i.e proportional amount was taken from each class for validation set).Hyper parameters were tuned for best results on validation set.

Similar class Categorizing effect Using a trained ConvBonsai Tree on CIFAR-10, Mnist-10 datasets with a depth of two and each leaf with a bonsai tree of depth 2 we found sub classification categories that were discovered are:
Which can be argued to have some latent classification and having more explainability than conventional CNNs.

Convolution leaf	**Sub categories CIFAR-10**	**Sub categories Mnist-10**
1	Bird, Deer, Frog	3,5,8
2	Cat, Dog, Horse	1,7,9,4
3	Car, Ship Truck	0,5,6
4	Bird, Aircraft	2,5

Table 2: Observed categorizing effect on second level of tree

Accuracy vs Memory comparison of different models :

We analyzed some standard and some resource efficient models on different benchmark dataset and observed accuracy achieved for different memory constraints.

Model	Values	IRIS		MNIST		USPS		notMNIST		CIFAR		MADELON	
		1	2	1	2	1	2	1	2	1	2	1	2
Bonsai	Size(kB)	0.1	0.2	0.836	47.34	0.86	6.144	5.25	52.46	31.46	659.48	10.37	40.07
	Accuracy	0.86	0.999	3.62	0.91	0.771	0.9347	0.899	0.931	0.109	0.3	0.56	0.55
Convolutional Bonsai	Size (kB)	NA	NA	3.94	8.035	NA	NA	2.11	31.33	8.5	63.5	NA	NA
	Accuracy			0.9248	0.91			0.7414	0.91	0.29	0.6		
ProtoNN	Size(kB)	0.2	0.74	31.03	164.84	56.05	97.07	51.53	210.45	1210.74	1242.96	98.67	107.8
	Accuracy	0.342	0.895	0.8048	0.8268	0.9063	0.9193	0.8889	0.8868	0.0984	0.0986	0.5	0.5
BNC	Size(kB)	0.02	0.025	6	23.83	0.61	10.22	6	23.43	14.13	39.125		
	Accuracy	0.605	0.657	0.7348	0.85	0.517	0.828	0.6934	0.7903	0.4348	0.2459		
kNN	k	1	5	1	17	1	5			1	5	1	5
	Accuracy	0.973	0.9602	0.9926	0.9852	0.9703	0.9537			0.815	0.273	0.83	0.759
SVM(linear)	Size(kB)	0.1		7.63		4.49		25.48		16.36		0.74	
	Accuracy	0.9726		0.908		0.9123		0.3457		0.201		0.481	
SVM(rbf)	Size(kB)	0.14		3.90		4.644		12.67		3.90		0.78	
	Accuracy	0.97		0.1008		0.8968		0.1627		0.1885		0.5	
D Tree	Size(kB)	0.07											
	Accuracy	0.94											
RFC	Size(kB)	2.01		2.73		196.11		754.25		526.6		2.73	
	Accuracy	0.94		0.7988		0.84		0.92		0.25		0.74	

Table 3: Empirical evaluation of different models based on memory requirements vs accuracy.

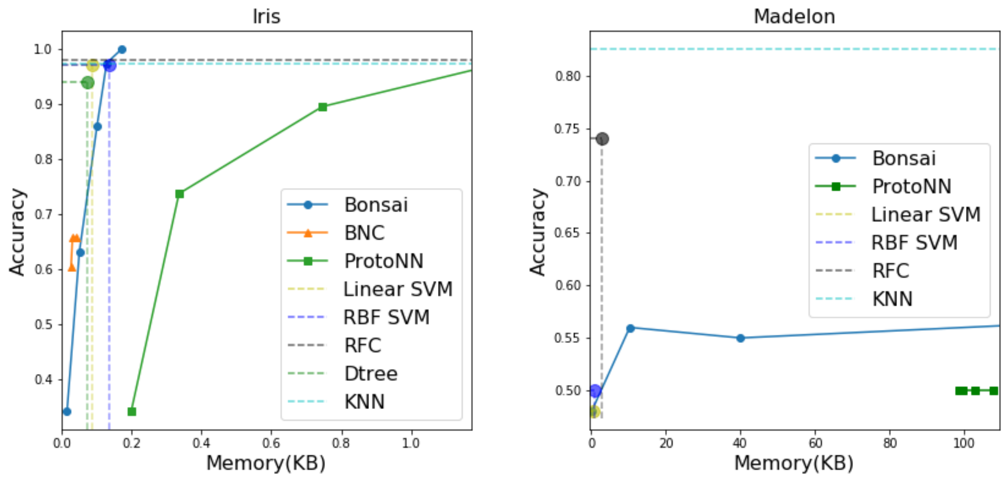

Figure 1: Comparison of different models on standard datasets

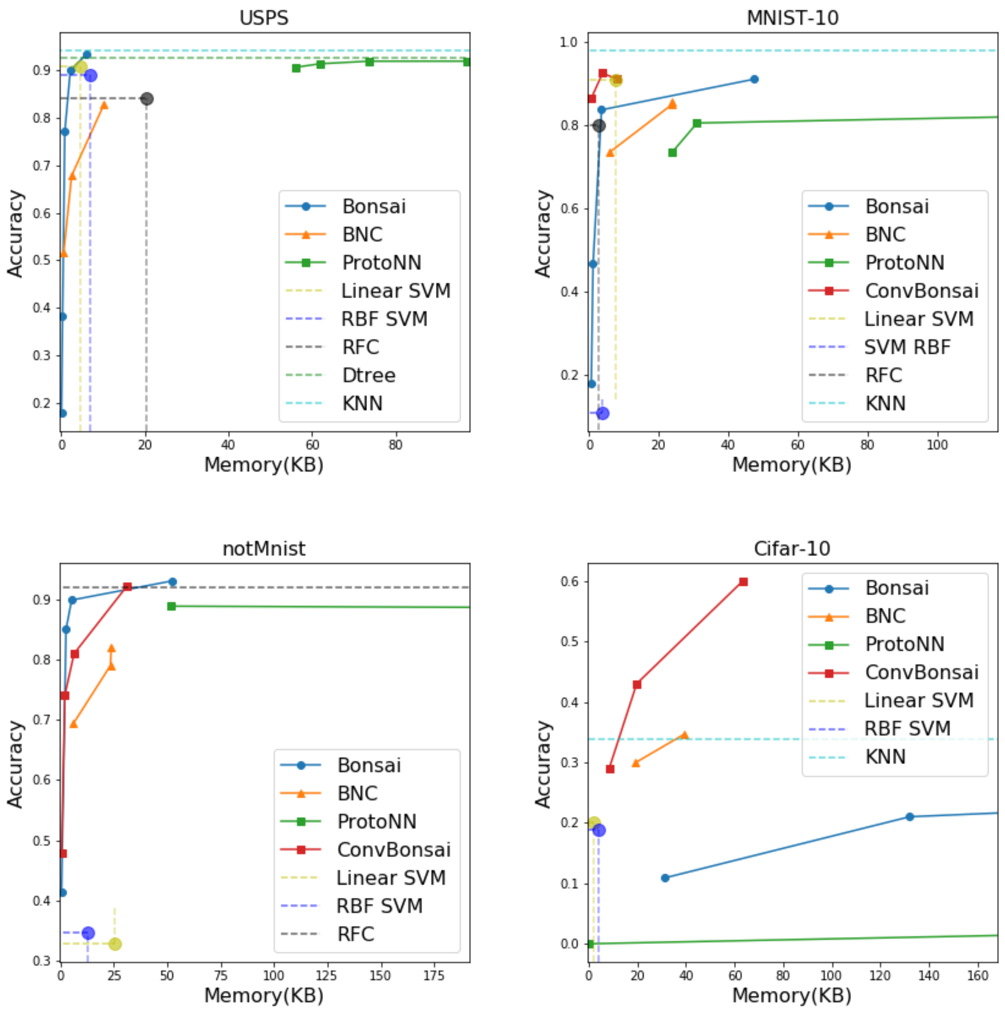

Figure 2: Comparison of different models on standard datasets

6 Conclusion and Future Possibilities

This project analyzes how different models developed for resource constraint devices work, and how they perform on different benchmark datasets. This project also proposes a convolutional modification of bonsai tree which helps in classifying data with local structure using less number of parameters, also it is more explainable than standard CNNs, comparison between different methods across multiple classification datasets show that ConvBonsai over performs all other methods for some datasets with nonlinear decision boundaries in terms of accuracy achieved for a given constraint on resource but other models like BNC performs better when the model decision boundaries can be linear like iris dataset which has good set of features. Also Bonsai works better as compared to methods like BNC(Binary neighbour compression) and ProtoNN for limited resources. we also conclude that these models can be trained on a laptop and can be transferred to IoT devices satisfying resource requirements of model.

Future Possibilities

There are lot of possibilities in this filed of research on how to develop compressed models that can learn good feature representation with as less memory as possible. some of the ways that can be further done based on this project are:

- Extending this ConvBonsai and Bonsai for Online Learning.
- Further Compressing the convolutional filters by pruning methods.
- Feature importance extraction using tensor decomposition from these models.
- Showing that ConvBonsai can learn new class representation without forgetting the old classification task.
- A new tree like model for RNN networks can be proposed.

further modification can be thought of and this whole area can be expanded to move towards a more explainable as well as compressed era of machine learning models where simplicity will be efficiency.

Acknowledgments

We are very thankful to Prof. Piyush Rai for his invaluable suggestions and discussions related to the our project. His directions, feedback and help throughout this project for designing the algorithm were indispensable.

References

[1] Chirag Gupta , Arun Sai Suggala, Ankit Goyal, Harsha Vardhan Simhadri, Bhargavi Paranjape, Ashish Kumar, Saurabh Goyal Raghavendra Udupa, Manik Varma, Prateek Jain, ProtoNN : Compressed and Accurate kNN for Resource-scarce Devices, Proceedings of the 34 th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017

[2] Kai Zhong, Ruiqi Guo, Sanjiv Kumar, Bowei Yan, David Simcha, Inderjit S. Dhillon, Binary Neighbor Compression, Fast Classification with Binary Prototypes, Proceedings of the 20 th International Conference on Artifi- cial Intelligence and Statistics (AISTATS) 2017

[3] Matt J. Kusner,Stephen Tyree, Kilian Weinberger, Kunal Agrawal, Stochastic Neighbor Compression, Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63130

[4] Ashish Kumar , Saurabh Goyal , Manik Varma , Bonsai Tree, Resource-efficient Machine Learning in 2 KB RAM for the Internet of Things, Proceedings of the 34 th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

[5] Feng Nan,Joseph Wang,Venkatesh Saligrama, Pruning Random Forests for Prediction on a Budget, 30th Conference on Neural Information Processing Systems (NIPS 2016)

[6] Cristian Bucila, Rich Caruana, Alexandru Niculescu-Mizil, Model Compression, KDD06, August 2023, 2006

[7] S. Han, H. Mao, and W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding, in Proc. Int. Conf. Learning Representations, 2016.

[8] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, Low-rank matrix factorization for deep neural network training with high-dimensional output targets, in Proc. IEEE Int. Conf. Acoustics Speech Signal Processing, 2013

[9] Y. Wang, C. Xu, C. Xu, and D. Tao, Beyond filters: Compact feature map for portable deep model, in Proc. 34th Int. Conf. Machine Learning, 2017

[10] Deboleena Roy, Priyadarshini Panda, Kaushik Roy , Tree-CNN: A Hierarchical Deep Convolutional Neural Network for Incremental Learning, arXiv:1802.05800 [cs.CV], 2018.

[11] Dmitry Laptev, Joachim M. Buhmann, Convolutional Decision Trees for Feature Learning and Segmentation. Pattern Recognition: 36th German Conference, GCPR 2014, Mnster, Germany, September 2-5, 2014, Proceedings

[12] Vikas K. Garg, Lin Xiao, Ofer Dekel, Sparse Multi-Prototype Classification, 2017

[13] Our code for ConvBonsai : `https://github.com/scakc/ConvBonsai-Tree/blob/master/ConvBonsai.ipynb`

[14] Project Repository : `https://github.com/yadavdeepak95/Pocket-ML`